



Documentando APIs RestFul com





Quando falamos de documentação de APIs é preciso ter em mente que tal artefato engloba tanto necessidades de negócio quanto necessidades técnicas. Por um lado, a documentação de uma API pode conter a elucidação de algumas regras de negócio, facilitando assim o seu entendimento pelas partes interessadas. Por outro lado, conterá detalhes técnicos, como os endpoints disponíveis, os parâmetros de entrada das requisições, suas respostas, possíveis erros, entre outros dados. Além disso, é preciso ter em mente quem acessará a documentação ou a quem ela será endereçada: qualquer pessoa, como pessoas de negócio ou outros desenvolvedores, fará uso dessa documentação para entender como usar a API que você desenvolveu. Logo, a documentação deverá ser completa o suficiente para permitir que pessoas externas à sua organização, ou que pessoas diferentes daqueles que desenvolveram a API ou que participaram, de alguma forma, do seu desenvolvimento consigam entender como ela funciona e como ela poderá ser utilizada. Inclusive, novos colaboradores que venham a trabalhar na API em questão, também poderão fazer uso dessa documentação, assim como desenvolvedores de outros times ou de outros produtos, numa mesma organização, como as equipes de front-end ou mobile. Além de todas essas funcionalidades, há ainda uma outra bastante útil e disponível na maioria das ferramentas de documentação: possibilidade de testar a API, de realizar requisições, testando assim todos os endpoints e recursos disponíveis.



A partir da introdução acima, podemos compreender um pouco a importância de documentarmos uma API. Tal processo, [illegible]n a utilização de ferramentas específicas, e ao contrário do que possa parecer, é um processo simples. Ao longo desse módulo conheceremos algumas ferramentas para a documentação de APIs Restful e que podem ser utilizadas juntamente com o Spring – direta (com fácil integração) ou indiretamente (independentes do framework). Além disso, veremos como incluir a biblioteca Springdoc em nossas APIs.

A escolha de uma ferramenta para documentar uma API passa por alguns fatores, como facilidade de instalação, configuração e uso, do ponto de vista do desenvolvedor; quantidade de recursos disponibilizados; usabilidade e design, por parte do usuário final, seja ele técnico ou não, entre outros. A seguir, serão listadas algumas ferramentas, dentre as mais utilizadas no mercado. Cada uma delas conta com características próprias, embora, ao final, todas cumpram o mesmo papel.

- Swagger: possivelmente a ferramenta mais conhecida e mais utilizada. Trata-se de uma ferramenta simples de se instalar e utilizar e que fornece uma grande quantidade de opções de customização [https://swagger.io/tools/swagger-ui] ;

- Springdoc / OpenApi: bastante similar ao Swagger e voltada especificamente para aplicações escritas com Spring Boot (diferentemente do Swagger e demais bibliotecas aqui listadas). Dada a sua facilidade de integração à API – como veremos mais adiante na prática, vem ganhando cada vez mais usuários [https://springdoc.org/]; **(Acesso em 28/09/2022)**

- DapperDox: Possui visual um pouco mais rebuscado que as opções vistas até aqui, além de permitir a customização de tema, entre outras funcionalidades. Como as demais bibliotecas, é uma solução Open-Source [https://dapperdox.io]; **(Acesso em 28/09/2022)**

- OpenAPI Generator: Destaca-se por suportar diferentes tipos de integração e casos de uso, possuindo, para isso, vários plugins [https://openapi-generator.tech/]; **(Acesso em 28/09/2022)**

- Redocly: Trata-se de um produto comercial, possuindo também uma versão gratuita, limitada a um desenvolvedor. Entre as opções vistas anteriormente, destaca-se pela riqueza de sua interface e recursos disponíveis na documentação gerada através dela [https://redocly.com/redoc/]. **(Acesso em 28/09/2022)**

Antes de prosseguir, visite os sites acima, leia um pouco mais sobre cada ferramenta e, sobretudo, visualize os exemplos de documentação disponíveis em cada um deles. Isso te permitirá ter uma melhor compreensão sobre os diferentes tipos de documentação e opções disponíveis.

A partir de agora, veremos como criar a primeira versão da documentação de uma API. Para isso, precisaremos de uma API (previamente criada). O passo seguinte consiste na instalação da dependência springdoc, no pom.xml. Edite o arquivo e inclua o código abaixo, na seção de dependências:

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.6.11</version>
  </dependency>
```



Pom.xml: Inclusão da Dependência Springdoc

OBS: atualmente, a versão mais atual disponível é a 1.6.11. Confira no site qual a versão mais atual, no momento em que for configurar a biblioteca em questão.

Já podemos visualizar a documentação inicial de nossa API. Coloque a API para rodar e, no navegador, acesse o endereço http://server:port/context-path/swagger-ui.html (normalmente, considerando as configurações default do spring, o endereço será: http://localhost:8080/swagger-ui/index.html). **(Acesso em 28/09/2022)**

OBS: a página referente ao endereço acima só estará online e disponível se você estiver rodando a API em seu computador.

A figura 1 mostra um fragmento da tela exibida ao acessarmos a documentação:

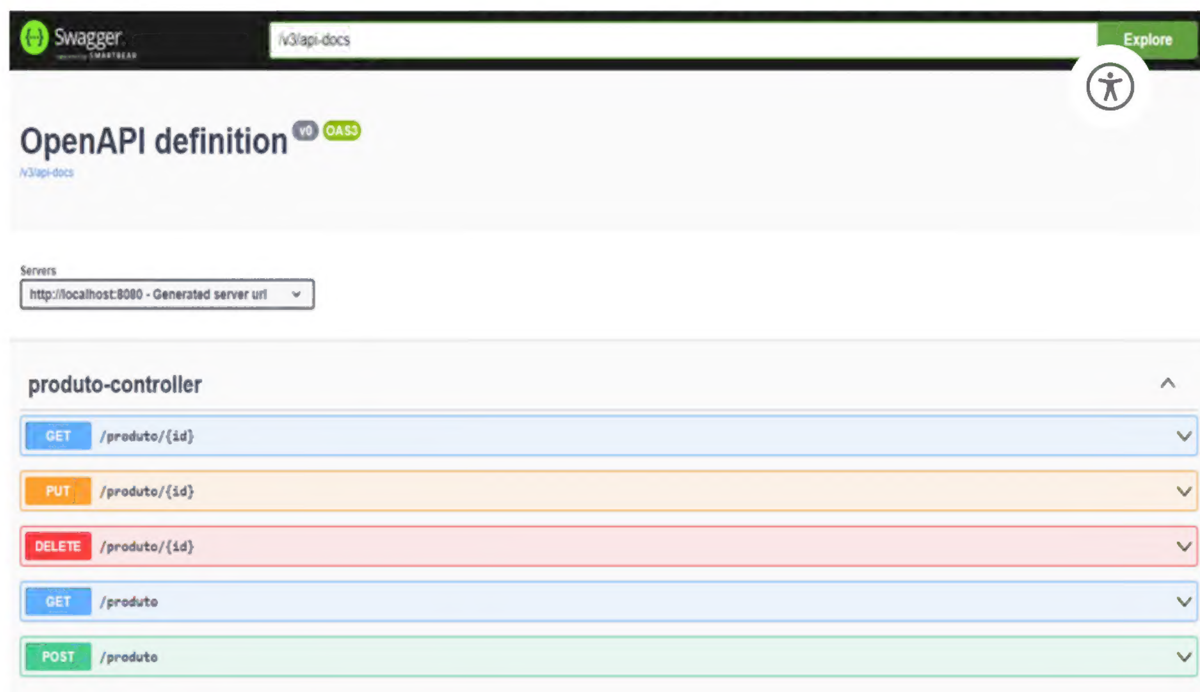


Figura 1: Documentação com Springdoc (Swagger UI)

Acima é possível ver alguns endpoints, abaixo de "produto-controller". Isso acontece, porque na API na qual a Springdoc foi configurada, tal endpoint encontra-se implementado. Caso não apareça nenhum endpoint para você, implemente alguns, suba novamente a API e atualize o Swagger UI. Feito isso, navegue pela documentação, clicando em cada endpoint. Teste as requisições e também suas respostas. Perceba que, mesmo numa implementação "crua", e bastante simples – basta inserirmos a dependência no pom.xml, e já é possível termos uma documentação bastante útil e também funcional. A seguir, veremos como customizar algumas das informações e dados exibidos na documentação.

Para começar, vamos editar algumas informações do cabeçalho – textos iniciais, como o título "OpenAPI definition", demonstrado na figura 1. Crie um novo package em seu projeto, chamado "config".

Dentro dele, crie uma classe chamada SpringdocConfig e inclua o código abaixo:



```
package br.com.descomplica.projeto.seguranca.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import io.swagger.v3.oas.models.ExternalDocumentation;
import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Info;
import io.swagger.v3.oas.models.info.License;

@Configuration
public class SpringdocConfig {
        @Bean
         public OpenAPI springShopOpenAPI() {
            return new OpenAPI()
                  .info(new Info().title("API Restful com Documentação")
                  .description("Exemplo de API Restful utilizando Springdoc para documentação.")
                  .version("1.0.0")
                  .license(new License().name("Apache 2.0").url("http://springdoc.org")))
                  .externalDocs(new ExternalDocumentation()
                  .description("Link do Repositório da Aplicação - APIRestful Documentation")
                  .url("https://github.com/endereco-repositorio-api"));
         }
}
```

Classe de Configuração SpringdocConfig

Após inserir a classe acima, modificando as informações a seu critério, suba novamente a API e atualize o Swagger-UI. Veja que os dados modificados a partir da classe de configuração referem-se ao cabeçalho – dados iniciais/gerais da documentação.

Agora, vamos ver como customizar os dados de um determinado endpoint: abra uma classe Controller de sua API e, com base no código abaixo, insira algumas anotações.

```
//demais codigos da classe

@Operation(summary="Listar todas as categorias", description  = "Listagem de Categorias")
@ApiResponses(value = {
            @ApiResponse(responseCode = "200", description = "A requisição foi executada com sucesso."),
            @ApiResponse(responseCode = "400", description = "Requisição Inválida"),
            @ApiResponse(responseCode = "403", description = "Você não tem permissão para acessar esse recurso."),
            @ApiResponse(responseCode = "404", description = "Recurso não encontrado.")})
public ResponseEntity<List<Categoria>> getAll(){
        //implementação do método
}

//demais codigos da classe
```



No exemplo acima, foram inseridas as anotações @Operation e @ApiResponse em um determinado método no Controller. Na documentação, essas anotações são refletidas na inclusão de uma descrição para o método em questão e também em informações customizadas para cada status HTTP que pode ser retornado pelo mesmo. Modifique sua API e visualize as alterações no Swagger UI.

Com a demonstração de algumas das várias propriedades disponíveis no Springdoc para personalização da documentação de uma API Restful, chegamos ao final deste módulo. Logo abaixo você confere o repositório contendo o código-fonte apresentado ao longo deste módulo.

Link:
https://github.com/FaculdadeDescomplica/pratica_integradora_tecnologias_disruptivas/tree/main/modulo6 **(Acesso em 28/09/2022)**

Atividade Extra

Para conhecer as propriedades disponíveis, além de outras informações sobre a biblioteca, recomendo que visite a página oficial do Springdoc:

Link: https://springdoc.org/ **(Acesso em 28/09/2022)**



Referência Bibliográfica

BAELDUNG. **DOCUMENTING a Spring REST API Using OpenAPI 3.0**, 2022. Disponível em: <https://www.baeldung.com/spring-rest-openapi-documentation >. **(Acesso em 28/09/2022)**

BAELDUNG. **SETTING Up Swagger 2 with a Spring REST API Using Springfox**, 2022. Disponível em: <https://www.baeldung.com/swagger-2-documentation-for-spring-rest-api >. **(Acesso em 28/09/2022)**

Ir para exercício